



TITLE:

# The derivational complexity of string-rewriting systems (Algebras, Languages, Algorithms in Algebraic Systems and Computations)

AUTHOR(S):

KOBAYASHI, YUJI

---

CITATION:

KOBAYASHI, YUJI. The derivational complexity of string-rewriting systems (Algebras, Languages, Algorithms in Algebraic Systems and Computations). 数理解析研究所講究録 2010, 1712: 140-147

ISSUE DATE:

2010-09

URL:

<http://hdl.handle.net/2433/170225>

RIGHT:

# The derivational complexity of string-rewriting systems

YUJI KOBAYASHI

*Department of Information Science  
Toho University, Funabashi 274-8510, Japan*

## 1 Derivational complexity

Let  $\Sigma$  be a (finite) alphabet and let  $\Sigma^* = \cup_{n \geq 0} \Sigma^n$  be the free monoid generated by  $\Sigma$ . A (string)-rewriting system  $R$  is a nonempty subset of  $\Sigma^* \times \Sigma^*$ . An element  $r = (u, v)$  in  $R$  is called a rule of  $R$  and written  $u \rightarrow v$ . Suppose that a word  $x \in \Sigma^*$  contains  $u$  as a subword, that is,  $x = x_1 u x_2$  with  $x_1, x_2 \in \Sigma^*$ , then we can apply the rule  $r$  to  $x$  and  $x$  is rewritten to the word  $y = x_1 v x_2$ . In this situation we write as  $x \rightarrow_r y$ . If there is some rule  $r \in R$  such that  $x \rightarrow_r y$ , we write  $x \rightarrow_R y$ , and we call the relation  $\rightarrow_R$  the one-step derivation on  $\Sigma^*$  by  $R$ .

A rewriting system  $R$  is *terminating* on  $x \in \Sigma^*$  if there is no infinite sequence of derivation:

$$x \rightarrow_R x_1 \rightarrow_R \cdots \rightarrow_R x_n \rightarrow_R \cdots$$

starting with  $x$ .  $R$  is *terminating* (or *noetherian*), if it is terminating on every  $x \in \Sigma^*$ .

The maximal length of a derivation sequence starting with  $x$  is denoted by  $\delta_R(x)$ . For  $x$  on which  $R$  is not terminating, we set  $\delta_R(x) = \infty$ . The function  $d_R : \mathbb{N} \rightarrow \mathbb{N} \cup \{\infty\}$  defined by

$$d_R(n) = \max\{\delta_R(x) \mid x \in \Sigma^n\}$$

for  $n \in \mathbb{N}$  is the *derivational complexity* of  $R$ .

We are interested in what functions can be derivational complexities of terminating finite rewriting systems.

Let  $\mathbb{R}_+ = \{x \in \mathbb{R} \mid x \geq 0\}$ . For two functions  $f, g : \mathbb{N} \rightarrow \mathbb{R}_+ \cup \{\infty\}$ , if there is a constant  $C > 0$  such that  $f(n) \leq C \cdot g(n)$  for any sufficiently large  $n \in \mathbb{N}$ , we write as  $f \leq O(g)$ . If moreover  $g \leq O(f)$ ,  $f$  and  $g$  are called *equivalent*, and written as  $f = O(g)$ .

A function  $f : \mathbb{N} \rightarrow \mathbb{R}_+ \cup \{\infty\}$  is *super-additive* if

$$f(m+n) \geq f(m) + f(n)$$

holds for any  $m, n \in \mathbb{N}$ . A super-additive function is non-decreasing. It is easy to see that the derivational complexity of a rewriting system is super-additive.

For an integer  $k \geq 1$ , a rewriting system  $R$  has polynomial (derivational) complexity of degree  $k$ , if  $d_R(n) = O(n^k)$ . Any (nonempty) rewriting system  $R$  has at least linear complexity, that is,  $d_R(n) \geq O(n)$ .

**Example 1.1.** Let  $k \geq 2$  and let  $\Sigma_k = \{a_1, a_2, \dots, a_k\}$ . For  $2 \leq \ell \leq k$  let

$$C_\ell = \{a_1 a_\ell \rightarrow a_\ell a_{\ell-1}, a_2 a_\ell \rightarrow a_\ell a_{\ell-1}, \dots, a_{\ell-1} a_\ell \rightarrow a_\ell a_1\}.$$

Define a system  $P_k$  on  $\Sigma_k$  inductively as follows.

$$P_2 = C_2 = \{a_1 a_2 \rightarrow a_2 a_1\},$$

and

$$P_k = P_{k-1} \cup C_k$$

for  $k \geq 3$ . Then,  $P_k$  has polynomial complexity of degree  $k$ .

A rewriting system  $R$  has *exponential complexity*, if there are constants  $C \geq D > 1$  such that

$$D^n \leq d_R(n) \leq C^n$$

for sufficiently large  $n \in \mathbb{N}$ . The one-rule system  $\{ab \rightarrow b^2 a\}$  has an exponential derivational complexity.

Due to [4], a derivational complexity exists in each level of the Grzegorz hierarchy of primitive recursive functions. Even the Ackermann's function is attained ([5]). Actually, a derivational complexity can exceed any recursive function (see Section 2). Many studies have been done about the derivational complexity of *term* rewriting systems under specific termination techniques (see [7] and the references cited there). Here we shall discuss the derivational complexity of *string* rewriting systems under a general situation.

## 2 Q-systems and Turing machines

In this article we only consider deterministic Turing machines. Let

$$M = M(\Sigma, Q, q_0, F, \delta)$$

be a  $k$ -tape Turing machine, where  $\Sigma$  is a tape alphabet,  $Q$  is a set of states,  $q_0$  is an initial state,  $F$  is a set of final states and  $\delta$  is a transition function. We assume that the tapes are one-way infinite and each head never moves to the left of the initial position.

Let  $\Sigma_b = \Sigma \cup \{b\}$ , where  $b$  denotes the blank symbol. The transition function  $\delta$  is a mapping from  $(Q \setminus F) \times \Sigma_b^k$  to  $Q \times (\Sigma_b \cup \{L, R\})^k$ , where  $L$  and  $R$  are the symbols for the right and left moves of the heads respectively. If for each  $i$  with  $1 \leq i \leq k$ ,  $x_i y_i$  is a word written on the  $i$ -th tape and the machine is looking at the leftmost letter of  $y_i$  in state  $q$ , then the  $k$ -ple

$$c = (x_1 q y_1, x_2 q y_2, \dots, x_k q y_k) \tag{2.1}$$

is a *configuration* of  $M$ . The size  $|c|$  of a configuration  $c$  in (2.1) is defined by

$$|c| = |x_1 y_1 x_2 y_2 \cdots x_k y_k|.$$

For  $x \in \Sigma^*$ , let  $\tau_M(x)$  be the number of steps taken until  $M$  halts when it runs with input  $x$  written in the first tape of  $M$ . The *time function*  $t_M : \mathbb{N} \rightarrow \mathbb{N} \cup \{\infty\}$  of  $M$  is defined by

$$t_M(n) = \max \{ \tau_M(x) \mid x \in \Sigma^n \}.$$

For a configuration  $c$ , let  $\tau'_M(c)$  be the number of steps taken until  $M$  halts when it starts with  $c$ . In particular,  $\tau_M(x) = \tau'_M(q_0 x, q_0, \dots, q_0)$  for  $x \in \Sigma^*$ . Define the *total time function* function  $t'_M : \mathbb{N} \rightarrow \mathbb{N} \cup \{\infty\}$  of  $M$  by

$$t'_M(n) = \max \{ \tau'_M(c) \mid c : \text{configuration of size } n \}$$

Clearly,

$$t'_M(n) \geq t_M(n)$$

for any  $n \in \mathbb{N}$ .

A *Q-system* is a finite rewriting system  $R$  over an alphabet

$$\Sigma = Q \cup \Sigma_1 \cup \Sigma_2 \cup \{\$ \} \quad (\text{disjoint union})$$

consisting of rules only of the form

$$\begin{aligned} vqu &\rightarrow v'q'u', & \text{or} \\ vqu\$ &\rightarrow v'q'u'\$, \end{aligned}$$

where  $q, q' \in Q, u, u' \in \Sigma_1^*$  and  $v, v' \in \Sigma_2^*$ .

A word  $x \in \Sigma^*$  is *admissible* (resp. *weakly admissible*), if it is of the form  $vqu$  with  $q \in Q, v \in \Sigma_2^*$  and  $u \in \Sigma_1^*\$$  (resp.  $u \in \Sigma_1^* \cup \Sigma_1^*\$$ ).

For a  $Q$ -system  $R$  and for  $n \in \mathbb{N}$ , define

$$ad_R(n) = \max \{ \delta_R(x) \mid x \text{ is admissible and } |x| = n + 2 \}$$

**Lemma 2.1.** *For a  $Q$ -system  $R$ , we have*

$$ad_R(n) \leq d_R(n + 2)$$

for any  $n \in \mathbb{N}$ . If  $ad_R$  is super-additive, then

$$d_R(n + 1) \leq ad_R(n)$$

for any  $n \in \mathbb{N}$ . If  $ad_R$  is equivalent to a non-zero super-additive function, then

$$d_R(n + 1) \leq O(ad_R(n)).$$

There is a natural way to simulate one-tape Turing machines by string-rewriting systems ([3]).

Let  $M = M(\Sigma, Q, q_0, F, \delta)$  be a one-tape Turing machine. Here,  $\delta$  is a mapping from  $(Q \setminus F) \times \Sigma_b$  to  $Q \times (\Sigma_b \cup \{L, R\})$ . We define a Q-system  $R_M$  associated with  $M$  as follows.  $R_M$  is a rewriting system on the alphabet

$$\Omega = Q \cup \Sigma_b \cup \bar{\Sigma}_b \cup \{\$, \}$$
 (disjoint union),

where  $\bar{\Sigma}_b = \{\bar{a} | a \in \Sigma_b\}$  is a copy of  $\Sigma_b$ , and consists of the rules:

$$\begin{array}{ll} qa \rightarrow \bar{a}q' & \text{for } \delta(q, a) = (q', R), \\ \bar{a}'qa \rightarrow q'a'a & \text{for } \delta(q, a) = (q', L), \\ qa \rightarrow q'a' & \text{for } \delta(q, a) = (q', a'), \\ q\$ \rightarrow \bar{b}q'\$ & \text{for } \delta(q, b) = (q', R), \\ \bar{a}q\$ \rightarrow qa\$ & \text{for } \delta(q, b) = (q', L), \\ q\$ \rightarrow q'a\$ & \text{for } \delta(q, b) = (q', a). \end{array}$$

for  $a, a' \in \Sigma_b$ ,  $q \in Q \setminus F$  and  $q' \in Q$ .

For a word  $x \in \Sigma_b^*$ ,  $\bar{x}$  denotes the word obtained from  $x$  by replacing every letter  $a$  in  $x$  by  $\bar{a}$ . Since one step of the Turing machine  $M$  just corresponds to one rewriting by  $R_M$  we have

**Lemma 2.2.** *It holds that*

$$\delta_{R_M}(q_0x\$) = \tau_M(x), \quad \delta_{R_M}(\bar{x}qy\$) = \tau'_M(xqy)$$

for  $x, y \in \Sigma_b^*$  and  $q \in Q$ .

**Corollary 2.3.** *We have*

$$d_{R_M}(n+2) \geq ad_{R_M}(n) = t'_M(n) \geq t_M(n)$$

for  $n \geq 0$ .

If  $R$  is finite and terminating, then we can compute  $d_R$  by tracing all the derivation sequences (see Section 4), and it is a recursive function. Actually it can exceed any recursive function.

**Corollary 2.4.** *For any recursive function  $f$ , there exists a finite terminating rewriting system  $R$  such that*

$$d_R(n) \geq f(n)$$

for any positive  $n \in \mathbb{N}$ .

### 3 Time functions and derivational complexity

As we have seen in the last section, derivational complexity is related to the time functions of Turing machines.

**Lemma 3.1.** (cf. [2], [6]) For any  $k$ -tape Turing machine  $M$  with time function  $f(n) \geq O(n)$ , there exists a one-tape Turing machine  $M'$  such that  $t_{M'}(n) = O(t'_M(n)) = O(f(n)^2)$ .

Suppose that  $f$  is the time function of a  $k$ -tape Turing machine  $M$  such that  $f \geq O(n)$  and  $f^2$  is equivalent to a super-additive function  $g$ . Let  $M'$  be the one-tape Turing machine Lemma 3.1. We have

$$t'_{M'}(n) = O(f(n)^2) = O(g(n)).$$

Let  $R$  be the  $Q$ -system associated with  $M'$ , then by Lemma 2.1 and Corollary 2.3, we see

$$d_R(n+2) \geq t'_{M'}(n) = ad_R(n) \geq O(d_R(n+1)).$$

It follows that

$$O(f(n-2)^2) \leq d_R(n) \leq O(f(n-1)^2).$$

Thus, we have

**Theorem 3.2.** *Let  $f(n)$  be a time function of a Turing machine such that  $f \geq O(n)$  and  $f(n)^2$  is equivalent to a super-additive function. Then there exists a finite rewriting system  $R$  such that*

$$O(f(n-2)^2) \leq d_R(n) \leq O(f(n-1)^2).$$

We say that a function  $f : \mathbb{N} \rightarrow \mathbb{N}$  is computable in time  $O(g(n))$ , if there exists a (deterministic) algorithm computing  $f(n)$  within time  $O(g(n))$ , more precisely, if there exists a multi-tape Turing machine which computes binary  $f(n)$  for given binary  $n$  with time function  $t_M(n) \leq O(g(n))$ .

**Lemma 3.3.** *If  $f : \mathbb{N} \rightarrow \mathbb{N}$  is a function such that  $f(n) \geq O(n^2)$  and the binary  $f(n)$  is computable in time  $O(\sqrt{f(n)})$  for binary  $n \in \mathbb{N}$ , then  $\lfloor \sqrt{f(n)} \rfloor$  is equivalent to a time function of a Turing machine.*

Combining this lemma with Theorem 3.1 we have

**Theorem 3.4.** *Suppose that a function  $f(n) \geq O(n^2)$  is computable in time  $O(\sqrt{f(n)})$  in binary and equivalent to a super-additive function. Then, there exists a finite rewriting system  $R$  such that*

$$O(f(n-2)) \leq d_R(n) \leq O(f(n-1)).$$

## 4 Computing the derivational complexity

Let  $R$  be a rewriting system on  $\Sigma$ . Consider a derivation sequence of length 2:

$$x = x'ux'' \rightarrow_R x'vx'' = y = y'u'y'' \rightarrow_R y'v'y'' = z,$$

where  $u \rightarrow v, u' \rightarrow v' \in R$ . This sequence is *left canonical*, if

$$|x'| < |y'u'|.$$

A sequence is *left canonical*, if every subsequence of length 2 of it is left canonical. In particular, a sequence of length  $\leq 1$  is left canonical.

**Lemma 4.1.** *For a derivation sequence of length  $n$  from  $x \in \Sigma^*$  to  $y \in \Sigma^*$ , there is a left canonical sequence from  $x$  to  $y$  of the same length  $n$ .*

For a derivational sequence

$$p : x_0 \rightarrow_R x_1 \rightarrow_R x_1 \rightarrow_R \cdots \rightarrow_R x_n,$$

we define a number  $L(p)$  by induction on  $n$  as follows. When  $n = 1$  and  $p : x_0 = x'_0 u x''_0 \rightarrow_r x'_0 v x''_0$  with  $r = (u \rightarrow v) \in R$ , define

$$L(p) = |x'_0 u| = |x_0| - |x''_0|.$$

Suppose that  $n \geq 2$  and

$$x_{n-2} = x'_{n-2} u' x''_{n-2} \rightarrow_{r'} x'_{n-2} v' x''_{n-2} = x_{n-1} = x'_{n-1} u x''_{n-1} \rightarrow_r x'_{n-1} v x''_{n-1} = x_n$$

with  $r = (u \rightarrow v), r' = (u' \rightarrow v') \in R$ . Then, define

$$L(p) = L(p') + |x'_{n-1}| - |x'_{n-2}| + |u| + K - 1,$$

where  $p'$  is the subsequence

$$x_0 \rightarrow_R x_1 \rightarrow_R \cdots \rightarrow_R x_{n-1}$$

of  $p$  and

$$K = \max \{ |u|, |v| \mid u \rightarrow v \in R \}.$$

**Lemma 4.2.** *For any derivation sequence  $p$  of length  $n$  ( $\geq 1$ ) starting with  $x \in \Sigma^*$  we have*

$$L(p) \leq (2K - 1)(n - 1) + |x|.$$

**Lemma 4.3.** *A left canonical derivation sequence  $p$  can be found by tracing at most  $L(p)$  letters in the words appearing in  $p$ .*

**Theorem 4.4.** *Let  $R$  be a finite rewriting system on  $\Sigma$  with derivational complexity  $f$ . Then, given  $n \in \mathbb{N}$ ,  $f(n)$  can be computed deterministically in time  $C^{f(n)}$  for some constant  $C > 1$ .*

## 5 Complexities of the forms $n^\alpha$ and $\alpha^n$

In this section we give the results that there are finite rewriting systems with derivational complexities equivalent to  $n^\alpha$  (and  $\alpha^n$ ), if the computational complexity of the real number  $\alpha$  is relatively low, but there are no such systems if the complexity of  $\alpha$  is high. The author has been inspired by the discussions in [8].

A real number  $\alpha > 0$  is *computable in time  $f(n)$* , if a binary rational approximation  $a/b$  ( $a, b \in \mathbb{N}$ ) of  $\alpha$  such that  $b \leq O(2^n)$  and

$$\left| \alpha - \frac{a}{b} \right| < \frac{1}{2^n}$$

can be computed in time  $f(n)$  (refer to [9] for computable real numbers). We denotes this rational  $a/b$  by  $\alpha[n]$ .

**Lemma 5.1.** *Let  $\alpha > 0$  be a real number computable in time  $O(f(n))$ . Then for an integer  $\nu$ , the function  $g_{\alpha,\nu}(n) = 2^{\lfloor \alpha(\lceil \log_2 n \rceil - \nu) \cdot n \rfloor}$  is equivalent to  $2^{\alpha n}$  and can be computed in time  $O(f(\lceil \log_2 n \rceil - \nu) + n)$ .*

**Theorem 5.2.** *Let  $\alpha \geq 2$  be a real number computable in time  $(O(C^{2^n}))$  for some constant  $C > 1$ . Then, there is a finite rewriting system  $R$  with derivational complexity equivalent to  $n^\alpha$ .*

Next, we consider the exponential function  $\alpha^n$ . Because it is not super-additive, we need the following

**Lemma 5.3.** *Let  $\alpha > 1$  be a real number, then the function  $f_\alpha$  defined by*

$$f_\alpha(n) = \begin{cases} \alpha^n & \text{if } n \geq 1/\log \alpha \\ (e \log \alpha) \cdot n & \text{if } 0 \leq n < 1/\log \alpha \end{cases}$$

*is super-additive.*

The computational complexities of  $\alpha$  and  $\log_2 \alpha$  are closely related.

**Lemma 5.4.** *Let  $\alpha (> 1)$  be a real number computable in time  $O(f(n))$ . Then,  $\log_2 \alpha$  is computable in time  $O(f(n+2) + 4^n n^2)$ , and  $2^\alpha$  is computable in time  $O(f(n + \lceil \alpha \rceil + 2) + 8^n n^2)$ .*

If we use a faster algorithm to compute the product of two integers, for example, Schönhag-Strassen's algorithm (see [1]), we can improve Lemma 5.4, but this is enough for our purpose.

**Theorem 5.5.** *If a real number  $\alpha > 1$  is computable in time  $O(C^{2^n})$  for some constant  $C > 1$ , then there is a finite rewriting system  $R$  with derivational complexity equivalent to  $\alpha^n$ .*

By our results we see that, for example, the functions  $n^\alpha$  ( $\alpha \geq 2$ ),  $\alpha^n$  ( $\alpha > 1$ ) and  $2^{\alpha n}$  ( $\alpha > 0$ ) for a rational (or more generally an algebraic) number  $\alpha$  are equivalent to the derivational complexities of finite rewriting systems. For a transcendental number  $\alpha$  with low complexity such as  $\pi$  and  $e$ , they are also equivalent to the derivational complexities.

Using Theorem 4.4, we can give the other direction as follows.

**Theorem 5.6.** *Let  $\alpha > 1$  be a real number.*

(1) *If there is a finite rewriting system with derivational complexity equivalent to  $n^\alpha$ , then  $\alpha$  is computable in time  $C^{C^{2^n}}$  for some constant  $C > 1$ .*

(2) *If there is a finite rewriting system with derivational complexity equivalent to  $\alpha^n$ , then  $\alpha$  is computable in time  $C^{2^n}$  for some constant  $C > 1$ .*



## References

- [1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, MA, 1974.
- [2] J.-C. Birget, Infinite rewriting systems and complexity, *J. Symbolic Comp.* 25 (1998), 759 – 793.
- [3] R. V. Book and F. Otto, *String-Rewriting Systems*, Springer, New York, 1993.
- [4] D. Hofbauer, Termination proofs by multiset path orderings imply primitive recursive derivation lengths, *Theoretical Computer Science* 105 (1992), 129 – 140.
- [5] D. Hofbauer and C. Lautermann, Termination proofs and the length of derivations, *RTA1989, LNCS 355* (1989), 167-177.
- [6] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, MA, 1979.
- [7] G. Moser, *Proof Theory at Work: Complexity Analysis of Term Rewrite Systems*, Habilitation thesis, Univ. Innsbruck, 2009.
- [8] M. V. Sapir, J. -C. Birget and E. Rips, Isoperimetric and isodiametric functions of groups, *Annals Math.* 156 (2002), 345-466.
- [9] K. Weihrauch, *Computable Analysis*, Springer, Berlin Heidelberg, 2000.